Refine Search

Search Results -

Terms	Documents
L4 same modeling	14

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L15

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Thursday, March 16, 2006 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
	DB=USPT; PLUR=YES; OP=OR		
L15	L4 same modeling	14	L15
L14	L12 same l4	0	L14
L13	L12 same dasd	0	L13
L12	wirelist or netlist	2423	L12
L11	L4 same simulat$	33	L11
L10	l1 same device same test$	21	L10
L9	L2 same test$	21	L9
L8	L2 same storage	17	L8
L7	L1 and l4	13	L7
L6	L4 same l1	0	L6
L5	L4 same l2	0	L5
L4	L3 or dasd	4185	L4
L3	direct adj1 access adj1 device	131	L3
L2	l1 same hardware	144	L2

L1	software adj1 represent$	843	L1

END OF SEARCH HISTORY

L15: Entry 11 of 14 File: USPT May 6, 1997

DOCUMENT-IDENTIFIER: US 5627990 A
**** See image for Certificate of Correction ****
TITLE: Management system for a hierarchical data cache employing preemptive cache track demotion and restaging to adapt to access patterns

Other Reference Publication (1):
Lim et al. "Direct Access Storage Device (DASD) Modeling and Validation", 1988, pp. 1024-1029.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L15: Entry 5 of 14 File: USPT Dec 14, 1999

DOCUMENT-IDENTIFIER: US 6003022 A
TITLE: Database execution cost and system performance estimator

Detailed Description Text (189):
The tool takes the above input and first estimates, for each SQL statement within each transaction, the amount of CPU time, FIG. 7, and the number of pages accessed synchronously and asynchronously from each table 623 and index 621, FIG. 6B. The tool, then, constructs a queuing network model, where the CPU is modeled as M/M/m server, as described in Computer Performance Modeling Handbook, Lavenberg, Stephen S., Academics Press, 1983, chapter 3 (page 94), herein incorporated by reference, and the DASDs are modeled as M/M/1 servers as described in Chapter 3, page 69 of the above reference, herein incorporated by reference. The queuing network model represents the CPU and the DASDs on which the tables and the indexes accessed by the queries in the transactions modeled reside. The queuing network model is solved to compute the CPU and device (i.e., disk, DASD) utilizations and the transaction and query response time, i.e., elapsed time.

Previous Doc Next Doc Go to Doc#

L15: Entry 12 of 14 File: USPT Jun 28, 1994

DOCUMENT-IDENTIFIER: US 5325505 A
TITLE: Intelligent storage manager for data storage apparatus having simulation capability

Detailed Description Text (520):
The DASD Performance Optimizer, DPO, module 106, is one of the tuning analysis routines of intelligent storage manager 103. The DPO module 106 uses Artificial Intelligence techniques, e.g. Expert Systems, as well as mathematical analysis and modeling to analyze the performance of the various data storage devices in the computer system/network and make recommendations of data set movements that improve performance and service levels delivered to computer users.

L15: Entry 2 of 14 File: USPT Jan 10, 2006

DOCUMENT-IDENTIFIER: US 6985842 B2
TITLE: Bidirectional wire I/O model and method for device simulation

PRIOR-PUBLICATION:

DOC-ID	DATE
US 20020169588 A1	November 14, 2002

Description Paragraph (15):
Computer system 200 preferably utilizes well known virtual addressing mechanisms that allow the programs of computer system 200 to behave as if they only have access to a large, single storage entity instead of access to multiple, smaller storage entities such as main memory 204 and DASD devices. Therefore, while operating system 222, HDL Modeling application 224, I/O model 226 are shown to reside in main memory 204, those skilled in the art will recognize that these programs are not necessarily all completely contained in main memory 204 at the same time. It should also be noted that the term "memory" is used herein to generically refer to the entire virtual memory of computer system 200.

Previous Doc Next Doc Go to Doc#

This software representation of real consumer devices provides the ability to support devices as yet unknown, and to support new and various underlying networks, protocols and standards. The overall goal of the system according to the invention is to become a complete solution for the home environment. To achieve this, the system's protocol must define user-interface conventions, device abstractions and networking protocols to enable devices, yet to be built, that provide complete end-to-end solutions of value to the end-user. The inventors believe that previous efforts to establish technologies which progress this aim have not been received well, simply because they have added cost to devices without yielding user-visible benefit. By providing complete solutions, customers will see the benefit and value, and the industry can again move forward and innovate upon a new platform.

Brief Summary Text (14):
The invention is especially relevant to consumer electronics in the home entertainment environment, but is not restricted thereto. For example, another example of a technical field that is increasingly being affected by the trend toward all-digital control and modular architecture is automotive applications. Car manufacturers and supply industries are putting more effort into digital control systems that allow the incorporation of sophisticated functionalities at affordable prices: electronically controlled gauges, anti-lock brake systems, electronic fuel injection, electronic motor management and emission control, electronic climate control, adaptive automatic transmissions, car audio equipment, navigation equipment etc. These functionalities have all different kinds of sophistication and operation. Manufacturing is considerably simplified if, at the assembly line, modules are installed whose functional interconnection relies on control via the software representation as proposed by the invention. The manufacturer does not have to make special reservations, regarding the electronic equipment, for each different combination of options. Further, interaction among these modular sub-systems brings the user-friendliness and efficiency of an automobile on a higher level.

Detailed Description Text (3):
FIG. 1 is a block diagram of a control system 100 according to the invention. System 100 comprises multiple consumer electronics devices 102, 104, 106, 108, 110, . . . , and 112. System 100 further includes one or more control means such as control means 114 and 116, in the example shown. Devices 102-108 are connected to control means 114. Devices 110 and 112 are connected to control means. Control means 114 and 116 are interconnected. System 100 and its protocol, discussed in more detail further below, provide the infrastructure to control the routing and processing of content data by controlling the function modes of devices 102-112. Control data controls the processing of content data, such as audio and video content, by devices 102-112. Control means 114 and 116 are also coupled to each other. Control of system 100 is explained below. Control is enabled by control means 114 and 116 having a software representation available of each device connected thereto.

Detailed Description Text (17):
Devices 106, 108, 110 and 112 belong to the class of intelligent consumer-devices. Devices 106-112 each provide a forward-compatible `plug-and-play` capability. In essence, each of devices 106-112 contains a software representation: an optional user-interface (see below under `Appler`) and a device abstraction (see below under `Abstract Device`), both compiled into machine-independent bytecode. A simple extension has been added to the native protocol of each of device 106-112 to enable another device to detect it, and retrieve the compiled bytecode. That other device can now execute the bytecode locally, and provide control of device 106-112. Preferably, the communication protocol used between the downloaded device abstraction and real device is private. The control protocol in system 100 defines the discovery and download protocol, and an execution environment for the downloaded bytecode. It also defines (by convention) a core set of functionalities that the device abstraction for a given type of device (e.g. `VCR`) should

implement, so that third party applications may take advantage of the capabilities of the device. The `chicken-and-egg` enhancement problem has been side-stepped, however. Since the user-interface has been downloaded with the device abstraction, they can implement enhancements over and above the core set, and `de facto` standards can arise without revising the standard and dealing with an `obsolete` installed base. This `definition by convention` is one of the large differences between the modus operandi of the computing industry and consumer electronics industry. It permits rapid innovation and a form of commercial and technical `Darwinism`.

Detailed Description Text (30):
The user-interface for an Abstract Device is referred to as a `Device Application` or an `Applet`, or just `application` in this text. FIG. 2 shows an Applet 204 for Abstract Device 202. The Applet may be downloaded from the associated Class C device, or it may already be residing at the Class D device. The Applet runs on Class D device 114 or 116 being used by the user interacting with system 100. The Applet may not be running on the same Class D device which is hosting the Abstract Device. For example, the Applet for the Abstract Device of device 108, which is hosted by Class D device 114, may be run on Class D device 116. This may be the case when Class D device 114 has no means, e.g., a display, to support the Applet for device 108 that requires a graphical user-interface, whereas Class D device 116 does have a display of its own or does have direct access to a device with a display, e.g., Class C device 110. For this reason, an Applet and the Abstract Device communicate through a messaging system, to make the actual location of the Abstract Device transparent to the Applet. For more details, see `Messaging` below. There is also a Registry, which is used to locate Abstract Devices by their attributes. See under `Registry` below for more details. In the example of FIG. 2, Applet 204 interfaces with a user-interface 206, e.g., of control means 114.

CLAIMS:

1. A consumer electronics system, comprising:

(a) a plurality of consumer electronics devices; and

(b) a task-driven control means interconnecting the consumer electronics devices for controlling interaction among the consumer electronics devices,

(c) the control means storing software representations of the interconnected consumer electronics devices and communicating with the consumer electronics devices via their respective stored software representations, and

(d) at least one of the consumer electronics devices storing its own software representation within itself and downloading its own software representation to the control means for storage in the control means automatically without consumer interaction when first being interconnected to the control means.

2. The system of claim 1, wherein each of the respective software representations stored in the control means comprises a device abstraction for representing the respective device at a same semantic level common to all of the software representations.

3. The system of claim 2, wherein at least one of the respective software representations comprises an application of a user-interface to the device abstraction, the application being run on the control means.

14. A method of enabling interaction among a multiplicity of consumer electronics devices, comprising the steps of:

storing a software representation of a consumer electronics device in the consumer electronics device itself;

interconnecting the consumer electronics device having the stored software representation of itself to a control means, the control means already storing a software representation in the control means that corresponds to each consumer electronics device already interconnected to the control means, each of the stored software representations in the control means interfacing between the control means and its corresponding consumer electronics device; and

automatically downloading the software representation stored in the consumer electronics device to the control means for storage in the control means without consumer interaction when the consumer electronics device having the stored software representation of itself is first interconnected to the control means.

15. The method of claim 14 wherein the consumer electronics device storing a software representation of itself is a home entertainment devices.

Previous Doc Next Doc Go to Doc#

L9: Entry 3 of 21 File: USPT Sep 20, 2005

DOCUMENT-IDENTIFIER: US 6948096 B2
**** See image for Certificate of Correction ****
TITLE: Functional random instruction testing (FRIT) method for complex devices such as microprocessors

Detailed Description Text (4):
Generally, a simulation software and a simulation model are utilized by a computer system (not shown) for computing the expected response of a complex device under test (DUT) 130. The simulation model may indicate a software representation of a complex device under test (DUT) 130, and may be written using hardware description languages such as Verilog or VHDL, and may be provided on a computer tangible medium, such as memory devices; magnetic disks (fixed, floppy, and removable); other magnetic media such as magnetic tapes; optical media such as CD-ROM disks, or via Internet downloads, which may be available for plug-in or download into an existing operating system (OS) for computing the expected response of a complex device under test (DUT) 130.

L9: Entry 9 of 21 File: USPT Mar 16, 2004

DOCUMENT-IDENTIFIER: US 6708290 B2
TITLE: Configurable debug system with wire list walking

CLAIMS:

9. A software development system, comprising: a memory storage system holding a software development tool program; a host computer connected to the memory storage system, the host computer operable to execute the software development tool program; an access mechanism for connecting to a hardware system, the hardware system being operable to execute an application program; and wherein the software development tool is operable to support a plurality of hardware system architectures by using a method for automatically determining capabilities of a first hardware system connected to a test port, the method comprising the steps of: accessing a first database corresponding to a first hardware system selected from the multiple hardware system architectures for use during a software debug session; creating a software representation for use by the software development system of a set of debug components of the first hardware system in accordance with the first database such that interconnections of the set of debug components are also represented; accepting a first request to the software development system for a debug job in which the request is comprised of at least a first action and one or more events that will trigger the action during execution of the application program; traversing the software representation of the set of debug components to determine if a first debug component required to execute the first action is present in the first hardware system and if a first set of debug components capable of detecting the one or more events are interconnected to the first debug component in such a way that the first action will be triggered when the one or more events occur; sending a debug command to the first hardware system via the test port to initialize a set of debug components selected by the step of traversing; and performing the debug job using the set of debug components selected by the step of traversing while executing the application program on the first hardware system.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L11: Entry 12 of 33 File: USPT Dec 5, 2000

DOCUMENT-IDENTIFIER: US 6157949 A
TITLE: Data placement on direct access devices for media servers with cyclic re-broadcast capability

Detailed Description Text (5):
The simulation property parameters in FIG. 6 and FIG. 7 are listed as follows. Illustratively, note that the direct access device is a magnetic disk:

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L11: Entry 13 of 33 File: USPT Nov 28, 2000

DOCUMENT-IDENTIFIER: US 6154719 A
**** See image for Certificate of Correction ****
TITLE: Logic simulation system and method

Detailed Description Text (84):
FIG. 41 shows the concept of conversational simulation. In FIG. 41, the simulator executes the logic circuit simulation while conversing with the designer. If the design is to be changed while a simulation is in progress, the circuit data information, which is an example of the simulation model read onto the main memory, is changed on the spot while the simulation continues. The contents of the circuit data information are the same as the contents of the simulation model stored in the simulation model storage section 14. However, the circuit data information only exists on main memory while the simulation is being executed; the simulation model is stored in an external memory device such as a DASD (Direct Access Storage Device) before the simulation starts. Here, the simulation model is not necessarily changed, because the circuit data information on main memory is changed directly.

Previous Doc Next Doc Go to Doc#

L11: Entry 18 of 33 File: USPT Nov 23, 1999

DOCUMENT-IDENTIFIER: US 5991757 A
TITLE: Method and system for searching an array for an array value

Brief Summary Text (7):
Although emulation permits programs written for a prior architecture to be executed within a state-of-the-art data processing system, the processing overhead required to emulate a prior architecture can degrade the performance of a data processing system to such an extent that the software development costs saved by emulating existing software are insignificant compared with the performance penalty incurred. For example, in a typical emulation, the addressing scheme of the emulated architecture and the data processing system are different. Therefore, the address of each instruction and its associated operands must be translated before the instruction can be emulated. The translation overhead for operands is particularly high when emulating instructions such as "store multiple" which access numerous addresses. In addition, when emulating a data processing system which utilizes a virtual address space, the DASD and main store address spaces are typically simulated separately. Thus, when an emulated program requires a new page of virtual memory to be paged in, in addition to retrieving the required page from the backing store, the emulator must transfer the data residing within the virtual page from the simulated DASD to the simulated main store, as would be performed in hardware by the emulated system. Furthermore, during normal execution of a program, a processor typically prefetches instructions that are likely to be executed in order to minimize instruction latency. However, when emulating a system which supports reetrant programming, an instruction handler routine is typically fetched only after the completion of the instruction handler routine utilized to emulate the previous instruction. Fetching is serialized in this manner during emulation since the emulation of a program instruction could possibly modify the following program instruction, thereby changing which instruction handler routine should be fetched.

Detailed Description Text (5):
According to the present invention, in addition to operating system 36, main memory 34 stores emulator 38. As will be understood by those skilled in the art, emulator 38 is a software interface between operating system 36 and programs written to execute under an incompatible operating system. Emulator 38 includes a number of instruction handlers, each comprising one or more instructions, which are utilized to emulate program instructions, such as those within program 46, that are incompatible with operating system 36. Furthermore, emulator 38 maps registers and other resources referenced by program 46 to hardware registers within CPU 30 or emulated data areas (e.g., simulated main store 37 and simulated DASD 39) within L1 cache 32 or main memory 34. As described in greater detail below, according to an important aspect of the present invention, when program 46 is emulated, emulator 38 accesses instructions within program 46 directly out of simulated DASD 39 and does not page the program instructions into simulated main store 37.

Detailed Description Text (10):
To minimize both address translation and paging overhead during emulation, main store 58 and DASD 60 are each simulated as segments within the address space of system unit 12. As such, simulated main store 37 and simulated DASD 39 can both be accessed identically, thus enabling emulator 38 to avoid paging portions of program 46 between simulated DASD 39 and simulated main store 37 when fetching emulated

instructions. More importantly, since all program objects are stored contiguously within a single segment (i.e., the segment allocated to simulated DASD 39), address translation overhead is eliminated when fetching emulated instructions within program 46. In addition, address translations occasioned by data accesses are limited to one per operand by not allocating data objects across a segment boundary of simulated DASD 39. For example, when emulating the 256-byte move instruction described above, only the starting addresses of the fetch and store operands are translated, limiting the maximum number of translations required for an instruction to 2.

Detailed Description Text (29):
As illustrated, the process begins at block 200 when emulator 38 receives a request to search the disk directory of simulated DASD. The process then proceeds to block 202, which depicts emulator 38 initializing selected sector pointer 254 and a key value pointer 258. Referring now to FIG. 11, there is illustrated a pictorial representation of disk directory 250 maintained by emulator 38 within the simulated DASD 39. Disk directory 250 has a first sector pointer 252, which points to the first sector to be searched, a last sector pointer 256, which indicates the last sector to be searched, and a selected sector pointer 254, which indicates the current sector being searched. In addition, key value pointer 258 specifies the key value to be compared to the searched key value. Returning to FIG. 10, the process proceeds from block 200 to block 202, which illustrates initializing first sector pointer 252 to point to the first sector within disk directory 250 and initializing last sector pointer 256 to point to the last sector within disk directory 250. The process then proceeds to block 204, which depicts dividing the sectors into two sets. The first set includes sectors up to and including the sector number equal to half of the smallest power of two greater than or equal to the number of sectors being searched. The second set includes the remaining sectors. Selected sector pointer 254 is then pointed to the first sector within the second set. For example, if disk directory 250 includes 14 sectors, selected sector pointer 254 is pointed to the ninth sector at block 204 since 16 is the smallest power of 2 greater or equal to 14.

Detailed Description Text (34):
As has been described, the present invention provides an improved data processing system and method for emulating a program written to execute under an operating system that is incompatible with the operating system utilized by the data processing system. In a first aspect of the present invention, instruction handlers utilized to emulate instructions recognized by the second operating system are aligned with cache line boundaries of the instruction cache in order to minimize instruction cache misses. According to a second aspect of the present invention, the data processing of the present invention emulates instructions directly out of a simulated DASD backing store in order to minimize emulation overhead. Furthermore, in a third aspect of the present invention, opcodes of emulated instructions are prefetched in order to maximize utilization of the execution pipeline of the processor. The present invention also delays handling interrupts until a user-selected number of branch or other non-reentrant instructions are executed in order to minimize emulation overhead. Finally, according to a fifth aspect of the present invention, the emulator of the present invention implements a modified binary search algorithm in order to efficiently search a multilevel disk directory.

Previous Doc Next Doc Go to Doc#

L11: Entry 15 of 33 File: USPT Aug 1, 2000

DOCUMENT-IDENTIFIER: US 6097565 A
TITLE: Repeatable runout free servo architecture in direct access storage device

Detailed Description Text (14):
The concept is implemented and evaluated on a 6000 TPI 3.5" DASD simulation model. The inverse filter is implemented with a proportional and derivative servo controller only (with no integral term) so that low frequency estimation errors are not adversely enhanced by the numerical operations required to produce RRO*. FIG. 7A and FIG. 7B show the conventional track following mode and RRO-free idle mode spectrum of actuator current. The nominal 6.6 mA RMS is reduced to 3.6 mA by this operation (45% reduction). Frequency spectrum shows reduction in current amplitude at frequencies around 500 Hz. The amount of power saving that is made possible by this scheme depends on the magnitude of RRO present. Larger RRO components will result in greater power savings. To identify limitations of the inverse filter-based RRO* estimation process two numerical experiments are conducted with a comprehensive dynamic simulation model. FIG. 8 and FIG. 8B show a conventional track-following transfer function used to estimate the RRO* from the originally servo-written in RRO*. As can be seen in FIG. 8B, the match is not exact, and it demonstrates the non-obvious nature of the inverse filter concept. The random processes associated with a disk file causes the servo-on PES to drift due to hysteresis and electronic noise in the feedback system. The conventional control (FIG. 8A) exaggerates the drift. As can be seen in FIG. 8B, the match between estimated RRO* and actual RRO* is not good, particularly at low frequencies. This is attributed to a complex interaction of actuator hysteresis and controller transfer function characteristics. However, when the open loop transfer function is modified to have crossover and roll-off characteristics as shown in FIG. 9A, the reconstruction of RRO* was accurate. The controller of FIG. 9A still has sufficient vibration rejection characteristics, but due to its transfer function characteristics, advantageously does not amplyfying the higher harmonics. For example, the gain at the fundamental frequency of 70 Hz. (the frequency of rotation of the disk) is 0 db in the controller of FIG. 9A, which is considerably lower than the 20 db or so of the controller of FIG. 8A.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L7: Entry 6 of 13 File: USPT Sep 28, 1999

DOCUMENT-IDENTIFIER: US 5959536 A
TITLE: Task-driven distributed multimedia consumer system

Abstract Text (1):
A control system comprises multiple consumer electronics devices and task-driven control means coupled to the devices for controlling an interaction among the devices. The control means acts on respective software representations of each respective one of the consumer devices. By encapsulating the variable complexity of the task within a software representation, it can be made as simple or as sophisticated as needed to bring the capabilities up to a common level. Since the level of interface is common to the devices, applications can uniformly manipulate devices which embody very different levels of sophistication.

Brief Summary Text (9):
To this end, the invention provides a control system comprising multiple electronic devices interconnected through a task-driven control means for controlling an interaction among the devices. The control means acts on respective software representations of each respective one of the devices. Preferably, at least a specific one of the electronic devices downloads its respective software representation to the control means, e.g., upon connection to the control means. Preferably, each of the respective software representations comprises a device abstraction for representing the respective device at a semantic level common to the representations and at least a particular one of the respective software representations comprises an application of a user-interface to the device abstraction, the application being run on the control means.

Brief Summary Text (10):
In order to be more specific, the invention provides in the field of consumer electronics a control system comprising multiple consumer electronics devices (e.g., a television receiver, a VCR, a CD-player, DVD-equipment, but also a telephone, lighting control system, thermostat, even a PC, etc.) interconnected through a task-driven control means for controlling an interaction among the devices. The control means acts or respective software representations of each respective one of the consumer devices.

Brief Summary Text (12):
The software representation operated upon by the control means hides the idiosyncrasies of the associated one of the real consumer devices. The software representation presents a more uniform interface for higher levels of software, similarly to device drivers in an operating system. By encapsulating the variable complexity of the task within a software representation, it can be made as simple or as sophisticated as needed to bring the capabilities up to a common level. Since the level of interface is common to the software representations, applications can manipulate, in the same manner, real devices which embody very different levels of sophistication. As to user-interface aspects, these have become task-driven, in contrast with the device-driven character in the conventional approaches. The user-interface has become customizable and extendible owing to the uniformity at the common level provided by the software representations.

Brief Summary Text (13):